COMPUTER SYSTEMS AND NETWORKS

Vol. 4, No. 1, 2022

DOI: https://doi.org/10.23939/csn2022.01.098

УДК 004.451.26:681.118.5

ПРИНЦИПИ ПОБУДОВИ ТА РЕАЛІЗАЦІЯ СИСТЕМНОГО МЕНЕДЖЕРА ВІДКЛАДЕНОГО ЗАПУСКУ ПРОГРАМ

Б. В. Марчак, О. Л. Лашко

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин
E-mail авторів: bohdan.marchak.ki.2018@lpnu.ua, oksana.l.lashko@lpnu.ua

У статті проводиться дослідження способів планування запуску різних програм на персональному комп'ютері. Розглянуто основні проблеми із роботою великих команд серед IT-спеціалістів, та загальні складності при відкладеному запуску необхідних програм.

Здійснено аналіз, який доводить, що сьогодні все більше IT-компаній переходять на віддалений режим роботи, а це супроводжує ненормований графік та роботу команди розробників у різних часових поясах, тому розробка та супровід програмного забезпечення для планування програмних завдань є доцільними та актуальними. У статті приділено велику увагу факторам, що виникають на більш загальному рівні у світі та всередині самих IT-компаній.

Метою статті є висвітлення основних аспектів проведеного дослідження та етапів створення спеціального програмного рішення, яке забезпечує розв'язок задачі узгодженої роботи колективу розробників програм. Зокрема, надає можливість задавати перелік програм для запуску, обирати власні умови запуску програм, відображати стан завдань у реальному часі. Крім того, виконувати паралельний запуск програм в окремих часових лініях та зберігати повний звіт про виконані завдання.

Ключові слова: планування, відкладений запуск, програми, C++, таймер.

Вступ

Розробка програм та маніпуляції із кодом супроводжують послідовність використання різних інструментів. Чи то особлива компіляція великого проєкту за допомогою необхідної послідовності виклику .exe, .bat або інших типів файлів, чи необхідність автоматизованого запуску різного типу процесів з зовсім різними умовами та особливою послідовністю – приносить досить великі незручності людям, які вирішили займатися такою справою. До цього можна додати ще й додаткові зовнішні умови: необхідність запустити якісь програми в конкретний незручний час, автоматизувати послідовність запуску низки процесів із додатковими умовами, та зберегти звітність їх виконання для подальшого аналізу. Це спричиняє особливі незручності сьогодні, коли все більше розробників працюють у віддаленому режимі, і більшість має ненормований час роботи.

Одним із можливих прикладів є робота системного адміністратора чи людини, котра, залежно від поставленої задачі, виконує запуск послідовності файлів чи сценарію. Дуже незручно, якщо таке планування може виникнути в час, який зовсім не підходить для тієї людини. Тому, якщо сценарій подій відомий наперед, і є інструмент, що може допомогти виправити таку ситуацію, тоді це викликає велике полегшення у роботі.

Іншим прикладом є робота команди, де кожного дня необхідно розвивати продукт і передавати проміжні його версії на тестування спеціальній команді. Розробка, скажімо, ігрового продукту залежить від: спільної роботи дизайнерів, які кожного дня добавляють в проєкт нові компоненти, що належать візуальній частині, та пророблюють файли із різними конфігураціями для коректного та логічного ігрового процесу; програмістів, які займаються постійним виправленням некоректного функціонування, додаванням розширеного функціоналу, та іншими додатковими

завданнями. Все це призводить до того, що майже кожного дня потрібна нова проміжна версія ігрового продукту, щоб команда тестування встигала перевіряти всі проблемні місця. Робочий день обмежений, і щось важливе може доробитися кимось пізніше, аніж почнеться збирання нової версії гри. Тому для вирішення цієї задачі необхідне рішення, яке допоможе сконструювати послідовність виконання будь-яких типів файлів з додаванням умов для їх запуску і з налаштуванням конкретного часу старту такої послідовності файлів у найсприятливіший для усіх час.

1. Аналіз останніх досліджень та публікацій

Сьогодні все більше IT-компаній переходять на віддалений режим роботи. Це пов'язано із великою кількістю факторів, що виникають, як на більш загальному рівні у світі, так і всередині самих компаній. Світові зміни змусили велику кількість людей змінити свій звичний спосіб життя в офісі на працю віддалено зі свого дому [1].

Це спричинило додаткові складності в роботі, оскільки тепер більша частина команди розробників знаходиться у різних часових поясах, і якщо виникає потреба у плануванні роботи якоїсь програми чи скрипту, що часто може залежати від членів усієї групи, можуть виникнути незручності у вигляді виконання роботи у вільний, а то і нічний час [2–3].

Для планування запуску завдання необхідно скористатися одним із можливих підходів до створення програмного таймера. У різних мовах програмування є інструменти для роботи із активним потоком, і зазвичай це функції, які приймають аргументом кількість мілісекунд для затримки активного потоку, з якого була запущена функція. Прикладом такої функції у мові програмування C++ є std::this_thread::sleep_for(). Перевагою такого методу є простота у використанні відносно кожного запущеного потоку, оскільки для затримки достатньо викликати одну функцію. Проте тут є і значні недоліки. Основний полягає в повному зупиненні роботи потоку, а це означає, що будь-які заплановані у ньому завдання будуть виконані тільки після завершення роботи затримки. Ця проблема може серйозно вплинути на роботу програми, особливо якщо вона використовує свій потік для оновлення користувацького інтерфейсу, що призведе до повного зависання програми [4].

Кращим є метод програмного таймера, який полягає у використанні конкретних алгоритмів для вираховування часу спрацювання без великого навантаження на процесор, і мінімальним впливом на роботу програми. Такий таймер може реалізовуватися системою сигналів напряму з операційної системи, чи з використанням ефективніших підходів до алгоритму постійної перевірки актуального часу [5].

Для перевірки коректності роботи програми у частині виконавчого модуля використано метод «білої скриньки» [6] , а для тестування візуального модуля – метод «чорної скриньки» [7].

2. Постановка задачі

Метою роботи є проєктування та імплементація системного менеджера відкладеного запуску програм, що забезпечить можливість задавати перелік програм для запуску, обирати власні умови запуску програм, відображати стан завдань в реальному часі; виконувати паралельний запуск програм в окремих часових лініях та зберігати повний звіт від виконаних завдань. При цьому кількість програм для запуску в одній часовій лінії повинна досягати 20, а кількість часових ліній має відповідати кількості ядер процесора. Потрібно передбачити підтримку операційної системи Windows 7/8/10/11 (x32/x64) та всіх системних сповіщень.

3. Проєктування системного менеджера

Першочерговим кроком при розробці програмного продукту є створення структурної схеми. За допомогою неї можна зобразити кількість модулів та їхню комунікацію.

Умовно програмний продукт можна розбити на два модулі, що залежать один від одного:

- комунікаційний візуальний модуль;
- фоновий (прихований) виконавчий модуль.

Спершу про комунікаційний візуальний модуль. Основною його задачею є надання користувачу можливості створити особливе завдання та наповнити його відповідними часовими

лініями, які містять послідовно обрані користувачем файли для запуску. Ще однією задачею модуля є збір та відображення результативних даних про стан запущеного завдання.

Фоновий виконавчий модуль має декілька режимів роботи: очікування запиту від комунікаційного візуального модуля на запис нового завдання у списку, яке отримує індивідуальний таймер запуску; передача результативних даних до візуального модуля для подальшої візуалізації. Всі отримані результати зберігаються у окремому файлі для подальшого збереження прогресу.

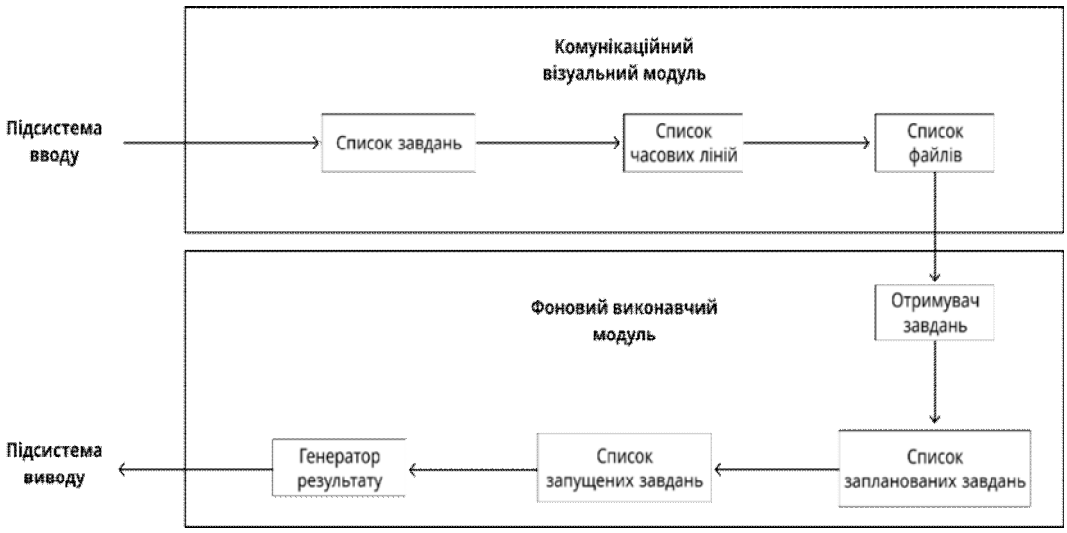

Рис. 1. Структурна схема проєкту

Розробка комунікаційного візуального модуля

При першому старті програми в операційній системі визначаються дані комп'ютера, щоб можна було до них прив'язати створення нових завдань і зберегти їх у файлі збереження, який можна знайти у папці програми за шляхом «Data/TaskList.xml». Із ним програма буде працювати, щоб зберігати весь актуальний прогрес.

Щоб додати нове завдання, користувач повинен натиснути на кнопку у вигляді символу «+», після чого на екран буде висвітлено віконце, в якому потрібно вписати назву для майбутнього завдання і натиснути кнопку «Add Task».

Сторінка «Manage Task» зберігає в собі таблицю результатів від попередньо запущеного завдання. Одне і те ж завдання можна запускати декілька разів і з різними параметрами та файлами. Кожен рядок таблиці містить в собі інформацію: хто запустив «Launched By», результат виконання «Status», хто створив «Creator», дату запису завдання у фоновий виконавчий модуль «Deploy Date», дату старту завдання «Start Date».

Основний функціонал для роботи із завдання забезпечує «режим редагування», який відкривається при натисканні на кнопку «Edit Mode». У цьому режимі графічно будується часова лінія між зображеним напрямом руху з «Begin» до «End». Додавання часової лініє відбувається за допомогою виклику контекстного меню на праву клавішу миші, і меню вибору «Add new pipeline».

На цьому етапі користувач може додати до створеної часової лінії файл для запуску, який буде додаватися у кінець списку. Клацнувши правою кнопкою миші, випадає контекстне меню із можливими способами додавання файлу: додавання пустого шаблону файлу для подальшого налаштування «Add New File», додавання одного існуючого файлу «Open File», чи декількох існуючих «Open Files…», а також опція видалення часової лінії «Remove Pipeline». Залежно від доданих файлів програма генерує картинку для полегшеного сприйняття користувача.

Усі часові лінії відображаються у вигляді вертикального списку, де кожен новий елемент додається до самого низу. Для полегшення візуального сприйняття часові лінії з'єднуються із вузлами «Begin» та «End», що символізує напрям руху виконання. Також ці лінії виконують роль індикатора, який фарбується у відповідний колір, враховуючи результат від приєднаної часової

лінії. Отже, якщо якась часова лінія буде мати загальну помилку, тоді вона пофарбується у червоний та передасть команду своїй індикаторній лінії на зміну кольору.

Також усі часові лінії мають оновлення свого стану в реальномі часі відносно статусу виконуваного завдання. Тож якщо одна часова лінія в цей момент запустилася, тоді фарбується у відповідний колір. У середині неї також в реальному часі відображаються всі зміни із внутрішніми файлами та кожним вузлом з'єднання й лініями. Тому користувач може в будь-який момент часу переглядати стан запуску кожного із файлів з точністю до того, через який саме файл згенерувалася помилка.

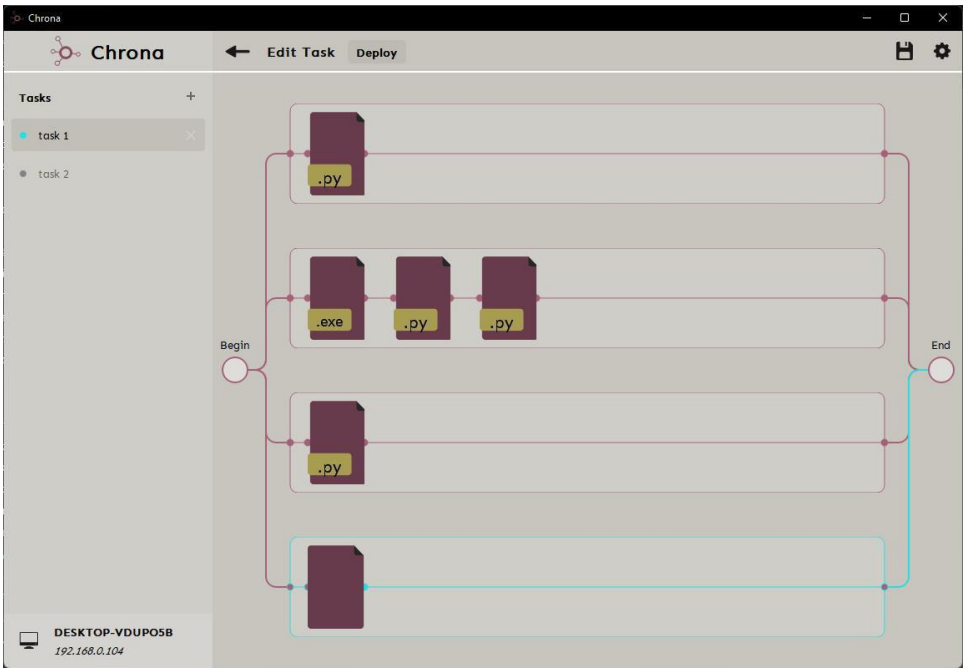

Рис. 2. Побудова і відображення паралельних часових ліній із різною кількістю файлів

Кожен доданий файл має можливість конфігурації своєї інформації, яку користувач може в будь-який момент до запуску змінити. При подвійному натисканні на якийсь файл відкривається віконце редагування «Edit File Info», в якому можна міняти дані про: шлях до файлу, «File Path», аргументи перед запуском програми «Pre_Run Arguments», аргументи запуску програми «Run Arguments», умову запуску: «After previous» якщо потрібно в будь-якому випадку виконати після попереднього; «If previous success» якщо потрібно виконати тільки у випадку, що попередній виконався успішно, і «If previous failed» якщо потрібно виконати тільки у випадку, що попередній виконався із помилкою.

Щоб передати завдання далі на очікування запуску, необхідно натиснути на кнопку "Deploy" і обрати необхідну дату та час.

Розробка виконавчого модуля

Основним класом, який керує всіма процесами у виконавчому модулі, є StartupManager. У його функціонал входить отримання інформації про завдання від візуального модуля, визначення готовності до роботи кожної часової лінії та виділення для них потоків виконання.

Після успішного аналізу завдання йому присвоюється індивідуальний таймер, який зберігається у списку таймерів, а сам таймер переводиться у режим відліку. Кожен такий таймер називається CustomTimer, і використовує механізм наслідування від класу QTimer, який забезпечує функціонал програмного таймера.

Сигнал від завершення таймера запускає функцію CatchTimerFinish, яка для кожної часової лінії у завданні виділяє свій клас потоку Starter, і переводить його у режим виконання.

Клас Starter призначений для виділення всіх необхідних ресурсів, щоб потік зміг надійно функціонувати і мав всі необхідні дані про часову лінію. Також кожне створення такого класу зв'язується сигналом void workFinished(Starter*) із головним класом StartupManager, щоб повідомити момент завершення своєї роботи.

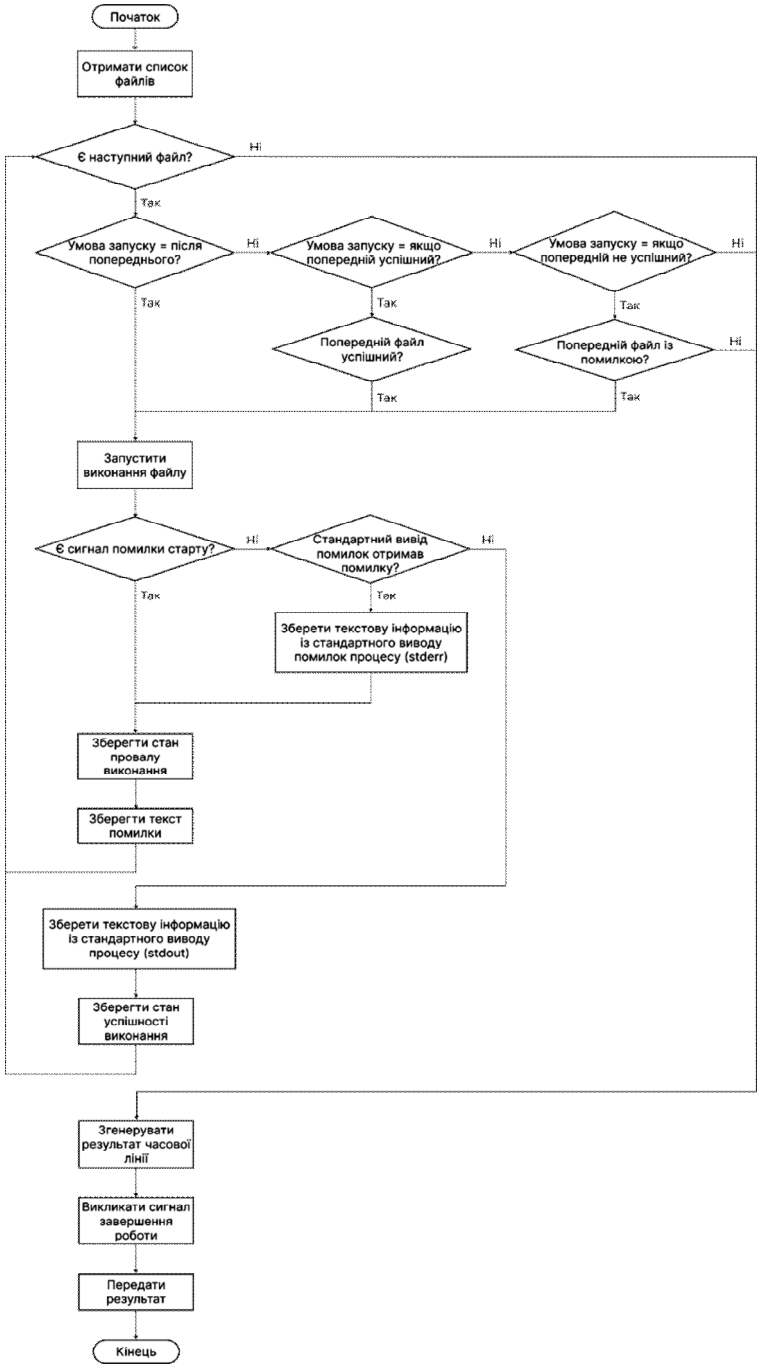

Рис. 3. Алгоритм опрацювання файлів у часовій лінії

Перед кожним запуском процесу Starter визначає необхідні умови, які потрібно порівняти із тими, що описані для кожного файлу окремо. Такими умовами є: запуск після попереднього файлу незважаючи на результат; запуск тільки якщо попередній виконався успішно; запуск тільки якщо попередній завершився із помилкою.

Після завершення виконавчого потоку відбувається сигнал до StartupManager у вигляді функції void CatchStarterWorkFinished(Starter *starter), який приймає результати виконання і пере-

віряє, чи всі поділені потоки для завдання завершилися. Якщо всі вони були завершені, тоді формується кінцевий результат, який надсилається на опрацювання до візуального модуля. Результат формується на основі кожного із завершених потоків, і якщо якийсь із них має помилку, тоді і кінцевий результат генерується як невдалий.

4. Перевірка працездатності розробленого системного менеджера

Для виявлення характеристик та можливостей, що надає створений програмний засіб, здійснено тестування виконавчого фонового модуля за методом «білої скриньки» [6] та комунікаційного візуального модуля – за методом «чорної скриньки» [7]. Результати представлено на поданих далі рисунках.

Зокрема, фрагмент на рис. 4. доводить коректність роботи виконавчого модуля при аналізі запланованого в часі завдання:

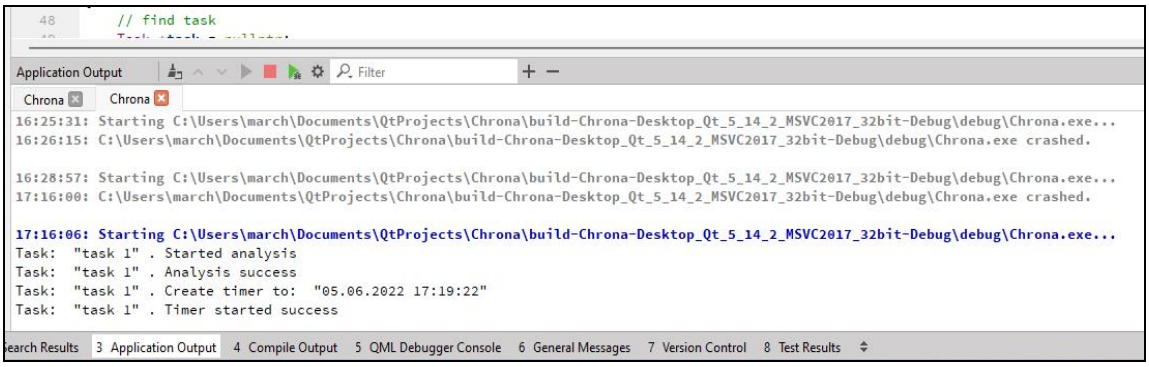

Рис. 4. Вивід виконавчого модуля про успішний аналіз завдання і створення таймера

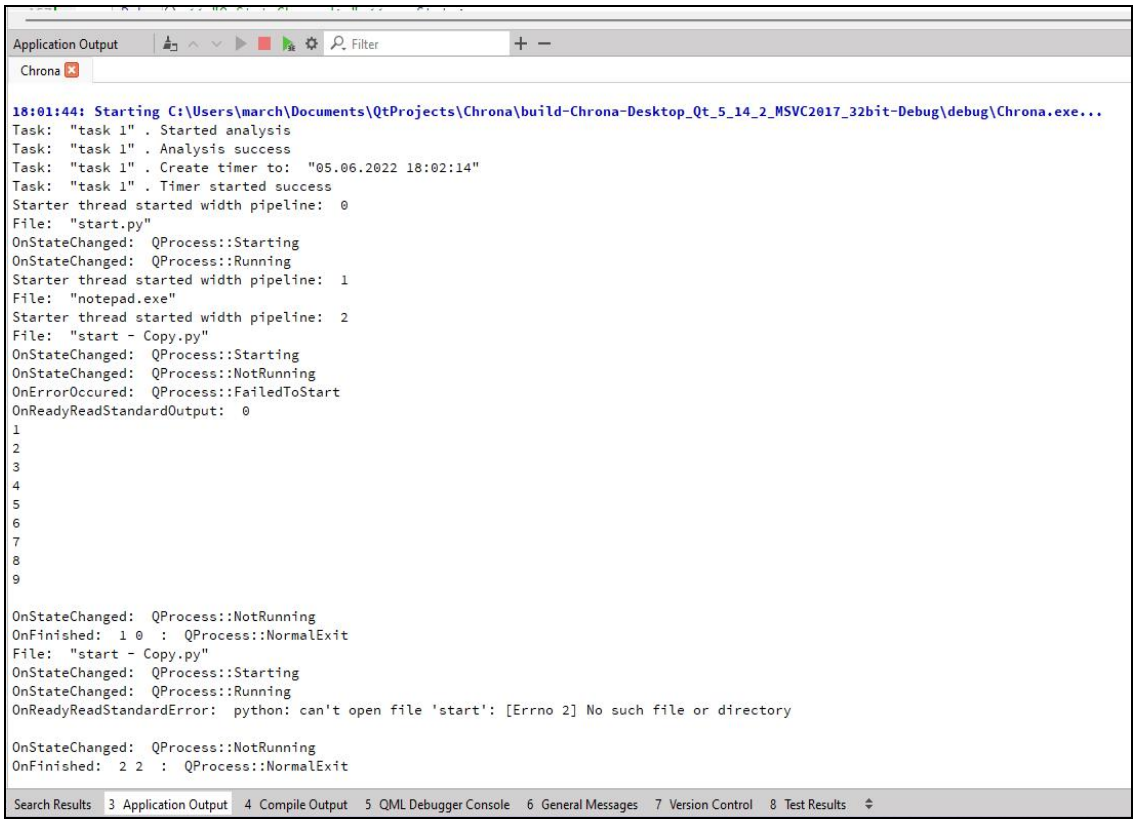

Рис. 5. Результат аналізу роботи трьох часових ліній

Наступний рисунок ілюструє можливість аналізу кількох часових ліній (рис. 5).

Далі маємо приклад візуального рішення для відображення різних завдань, який демонструє зрозумілість та інтуїтивне сприйняття інтерфейсу реалізованої програми (рис. 6).

Той самий результат можна переглянути у таблиці (рис. 7).

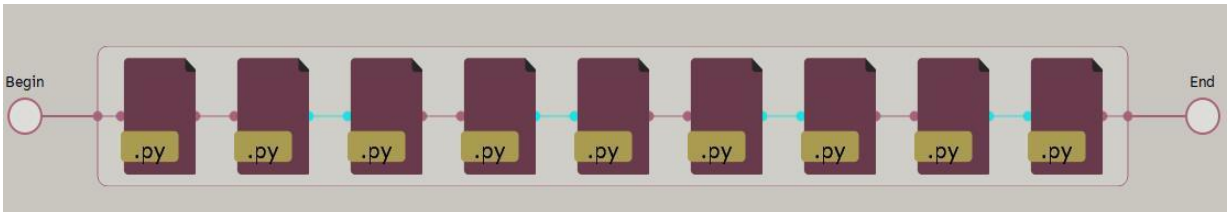

Рис. 6. Результат відображення часової лінії, де кожен другий – з помилкою

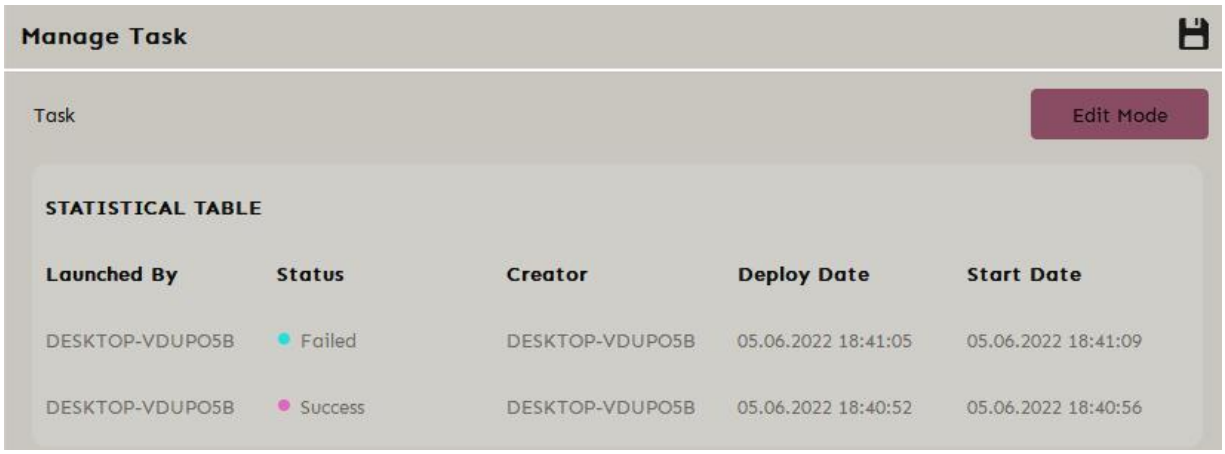

Рис. 7. Результат завдання із невиконаною умовою у таблиці

Усі кроки тестування однозначно показали адекватність створеного програмного засобу та відповідність його параметрів поставленим вимогам.

Висновки

Розглянуто спосіб планування запуску різних програм на персональному комп'ютері. Проведено аналіз теперішньої ситуації із IT-компаніями та їхніми працівниками, які здебільшого працюють віддалено. Окреслено основні проблеми у роботі розробників, що працюють командою у різних часових поясах. Сформульовано та розв'язано задачу узгодження асинхронної роботи членів команди шляхом відкладеного запуску їхніх файлів.

Розроблено програмний продукт, що дозволяє створювати заплановані завдання. Продумано та реалізовано алгоритм роботи виконавчого модуля, який надає можливість обирати умови запуску програм, дозволяє відслідковувати стан завдання у реальному часі. Досягнуто можливість паралельного запуску програм в окремих часових лініях та збереження повного звіту від виконаних завдань. При цьому кількість програм для запуску в одній часовій лінії досягає 20, а оптимальна кількість часових ліній відповідає кількості ядер процесора.

Створено зручний користувацький інтерфейс для більшої ефективності взаємодії користувача із програмним продуктом. Наголошено на основних функціях, які доступні користувачеві, та описано їхню результуючу дію.

Тестування системного менеджера довело, що він має низку якісних переваг над аналогічними продуктами на ринку,зокрема: не потребує окремого спеціаліста (системного адміністратора), який би створював скрипти запуску; не передбачає складного налаштування окремих запусків; розробка плану виконання файлів зводиться до роботи з елементами інтерфейсу програми, а не написання окремого коду. Це скорочує час на освоєння програмного засобу і тому збільшує

загальну ефективність команд, що користуються менеджером при роботі над спільним проєктом. Це, в свою чергу, веде до збільшення економічних винагород.

Отже, можна стверджувати, що розроблений програмний продукт має перспективу впровадження та подальшого розвитку своїх можливостей та розширення функціоналу відповідно до вимог ринку та відгуків користувачів. Загальні підходи, використані в розробці, можуть застосовуватися для створення широкого класу подібних засобів.

Список літератури

1. Yang L., Holtz D., Jaffe S. et al. The effects of remote work on collaboration among information workers. Nat Hum Behav 6, 43–54 (2022). DOI: 10.1038/s41562-021-01196-4 (accessed: 15 September 2022).

2. Toscano F. & Zappalà S. Overall job performance, remote work engagement, living with children, and remote work productivity during the COVID-19 pandemic: A mediated moderation model.European Journal of Psychology Open, 80(3), 133–142. (2021). DOI: 10.1024/2673-8627/a000015 (accessed: 5 Sept. 2022).

3. Lauring J., Drogendijk R., Kubovcikova A. The role of context in overcoming distance-related problems in global virtual teams: an organizational discontinuity theory perspective, The International Journal of Human Resource Management, (1), (2021). DOI: 10.1080/09585192.2021.1960584 (accessed: 15 September 2022).

4. Nicolai M. Josuttis. C++ Standard Library, The: A Tutorial and Reference, 2nd Edition. – Addison-Wesley Professional. 2012. P. 143 (accessed: 15 September 2022).

5. Lopes S. F., Vicente P. and Gomes R. "Library for simplified timer implementation using standard C++", 2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2015. Pp. 517–524 (accessed: 15 September 2022).

6. Muhammad Ali Gulzar, Shaghayegh Mardani, Madanlal Musuvathi, Miryung Kim, "White-box testing of big data analytics with complex user-defined function", ESEC/FSE 2019: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software EngineeringAugust 2019. 290–30,1. DOI: 10.1145/3338906.3338953 (accessed: 12 September 2022).

7. Jacob P. M. and Prasanna M. "A Comparative analysis on Black box testing strategies", 2016 International Conference on Information Science (ICIS), 2016. Pp. 1–6. DOI: 10.1109/INFOSCI.2016.7845290 (accessed: 15 September 2022).

PRINCIPLES OF BUILDING AND IMPLEMENTATION OF THE SYSTEM MANAGER OF POSTPONED PROGRAMS LAUNCH

B. Marchak, O. Lashko

Lviv Polytechnic National University,
Computer Engineering Department
Authors' e-mail: marchakboh@gmail.com, lashko.oksana@gmail.com

The article examines how to plan the launch of various programs on a personal computer. The main problems with the work of large teams among IT specialists, and the general difficulties in delaying the launch of the necessary programs are considered.

The analysis shows that now more and more IT companies are moving to remote mode, which is accompanied by irregular schedules and work of the development team in different time zones, so the development and maintenance of software for scheduling software tasks is appropriate and relevant. The article pays great attention to the factors that arise, both at a more general level in the world and within the IT companies themselves.

The purpose of the article is to highlight the main aspects of the study and the stages of creating a special software solution that provides a solution to the problem of coordinated work of the team of program developers. In particular, it provides the ability to set the list of programs to run, choose your own conditions for running programs, display the status of tasks in real time. In addition, run programs in parallel in separate timelines and keep a complete report of completed tasks.

Keywords: scheduling, delayed start, programs, C ++, timer.